Using differentiable simulations to tackle problems in fundamental plasma physics

Archis S. Joglekar

Ergodic LLC., Seattle, WA
Pasteur Labs, Seattle, WA
Laboratory for Laser Energetics, University of Rochester, Rochester, NY
University of Michigan, Ann Arbor, MI

2025 Plasma Kinetics Workshop

Collaborators & Acknowledgments

- Laboratory for Laser Energetics, Univ. Rochester
- Pasteur Labs
- University of Michigan, Ann Arbor

This material is based upon work supported by the DOE Office of Fusion Energy under Award Number(s) DE-SC0024863, NERSC award FES-ERCAP0026741, DOE NNSA under Award Number DE-NA0003856

From the perspective of scientific computing, hardware-accelerated AD is a key outcome of the machine learning revolution

- Training neural networks using gradient descent requires the acquisition of gradients of arbitrary numerical programs with respect to $\mathcal{O}(10^9)$ parameters. This can only be done using **Automatic Differentiation**

From the perspective of scientific computing, hardware-accelerated AD is a key outcome of the machine learning revolution

- Training neural networks using gradient descent requires the acquisition of gradients of arbitrary numerical programs with respect to $\mathcal{O}(10^9)$ parameters. This can only be done using **Automatic Differentiation**
- The tech companies have poured in $\$\mathcal{O}(10^6)$ to develop GPU-accelerated automatic differentiation libraries that have an easy to use, high-level API

From the perspective of scientific computing, hardware-accelerated AD is a key outcome of the machine learning revolution

- Training neural networks using gradient descent requires the acquisition of gradients of arbitrary numerical programs with respect to $\mathcal{O}(10^9)$ parameters. This can only be done using **Automatic Differentiation**
- The tech companies have poured in $\$\mathcal{O}(10^6)$ to develop GPU-accelerated automatic differentiation libraries that have an easy to use, high-level API
- From our perspective, these are just mathematical libraries with which we can build scientific programs. They give you AD and GPUs for free

From the perspective of scientific computing, hardware-accelerated AD is a key outcome of the machine learning revolution

- Training neural networks using gradient descent requires the acquisition of gradients of arbitrary numerical programs with respect to $\mathcal{O}(10^9)$ parameters. This can only be done using **Automatic Differentiation**
- The tech companies have poured in $\$\mathcal{O}(10^6)$ to develop GPU-accelerated automatic differentiation libraries that have an easy to use, high-level API
- From our perspective, these are just mathematical libraries with which we can build scientific programs. They give you AD and GPUs for free
- Example problems we can tackle using gradients of plasma physics simulations are

From the perspective of scientific computing, hardware-accelerated AD is a key outcome of the machine learning revolution

- Training neural networks using gradient descent requires the acquisition of gradients of arbitrary numerical programs with respect to $\mathcal{O}(10^9)$ parameters. This can only be done using **Automatic Differentiation**
- The tech companies have poured in $\$\mathcal{O}(10^6)$ to develop GPU-accelerated automatic differentiation libraries that have an easy to use, high-level API
- From our perspective, these are just mathematical libraries with which we can build scientific programs. They give you AD and GPUs for free
- Example problems we can tackle using gradients of plasma physics simulations are
  - **Closure modeling** - Minimize difference between kinetic and fluid simulations

From the perspective of scientific computing, hardware-accelerated AD is a key outcome of the machine learning revolution

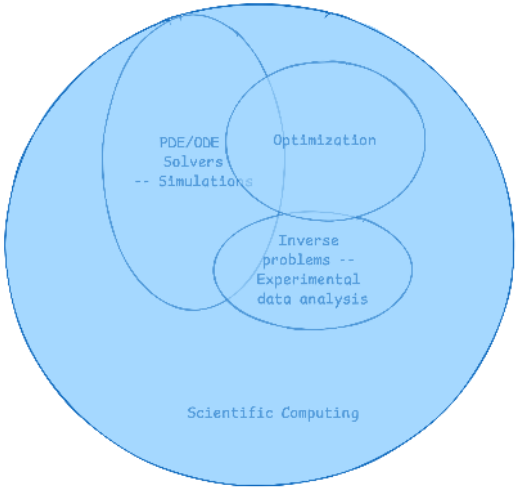
- Training neural networks using gradient descent requires the acquisition of gradients of arbitrary numerical programs with respect to $\mathcal{O}(10^9)$ parameters. This can only be done using **Automatic Differentiation**
- The tech companies have poured in $\$\mathcal{O}(10^6)$ to develop GPU-accelerated automatic differentiation libraries that have an easy to use, high-level API
- From our perspective, these are just mathematical libraries with which we can build scientific programs. They give you AD and GPUs for free
- Example problems we can tackle using gradients of plasma physics simulations are
  - **Closure modeling** - Minimize difference between kinetic and fluid simulations
  - **Dynamics discovery** - Maximize nonlinear activity of kinetic simulations

From the perspective of scientific computing, hardware-accelerated AD is a key outcome of the machine learning revolution

- Training neural networks using gradient descent requires the acquisition of gradients of arbitrary numerical programs with respect to $\mathcal{O}(10^9)$ parameters. This can only be done using **Automatic Differentiation**
- The tech companies have poured in $\$\mathcal{O}(10^6)$ to develop GPU-accelerated automatic differentiation libraries that have an easy to use, high-level API
- From our perspective, these are just mathematical libraries with which we can build scientific programs. They give you AD and GPUs for free
- Example problems we can tackle using gradients of plasma physics simulations are
  - **Closure modeling** - Minimize difference between kinetic and fluid simulations
  - **Dynamics discovery** - Maximize nonlinear activity of kinetic simulations
  - **Threshold discovery** - Maximize the 'specific heat' of the system (2^{nd} order derivative is needed!)

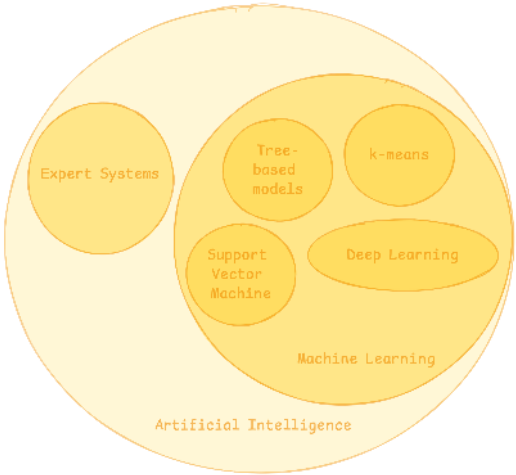
Scientific computing contains many useful paradigms for plasma physics

- ODE/PDE solvers form the bulk of our computational modeling tools
- Experimental data analysis often involves inverse problems



Deep Learning and Neural Networks are a subset of Artificial Intelligence

- Artificial Intelligence comprises algorithms that can perform tasks that require human level intelligence
- Machine Learning comprises algorithms informed by data rather than by humans
- Deep Learning is when the model is a neural network rather than e.g. Gaussian Processes



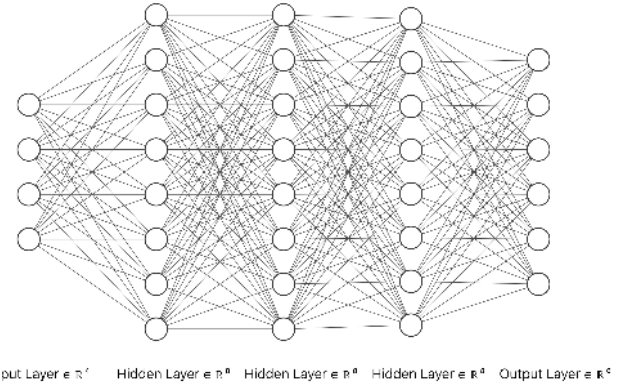
Deep Learning REQUIRES Automatic Differentiation

$$\vec{y} = f(\vec{x}) \qquad \hat{\vec{y}} = g(\vec{x};\theta)$$
$$\hat{\vec{y}} = A_4(\theta_4) \circ A_3(\theta_3) \circ A_2(\theta_2) \circ A_1(\vec{x};\theta_1)$$

$$A_1 : \mathbb{R}^4 \to \mathbb{R}^8 \qquad A_2 : \mathbb{R}^8 \to \mathbb{R}^8$$
$$A_3 : \mathbb{R}^8 \to \mathbb{R}^8 \qquad A_4 : \mathbb{R}^8 \to \mathbb{R}^6$$

$$\text{argmin } \mathcal{L}(\theta) : \mathcal{L}(\theta) = \sum [g(\vec{x};\theta) - f(\vec{x})]^2$$

$$\theta^{n+1} = \theta^n + \Delta\theta \frac{\partial L}{\partial \theta}$$



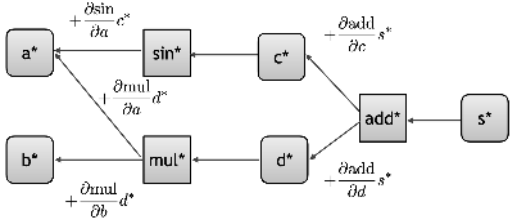
How do you get the gradient?

AD is effectively a numerical chain rule

- PyTorch, Tensorflow etc. have their own own math modules

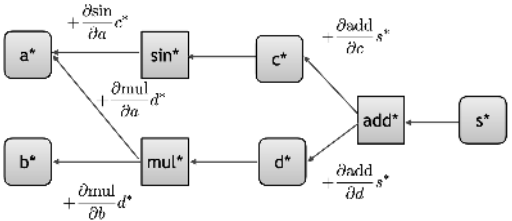
- PyTorch, Tensorflow etc. have their own own math modules
- In these modules, mathematical primitives have defined derivatives

- **Example:** $s(a, b) = \sin(a) + ab$



- PyTorch, Tensorflow etc. have their own own math modules
- In these modules, mathematical primitives have defined derivatives
- Code composed with these primitives has derivatives using the chain rule

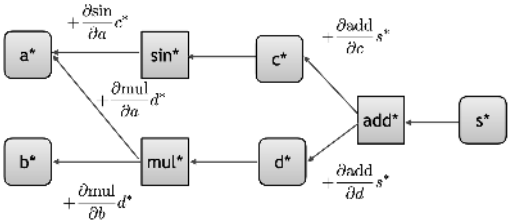
- **Example:** $s(a, b) = \sin(a) + ab$



```
from jax import grad
import jax.numpy as jnp

def tanh(x):  # Define a function
  y = jnp.exp(-2.0 * x)
  return (1.0 - y) / (1.0 + y)

grad_tanh = grad(tanh)  # Obtain its gradient function
print(grad_tanh(1.0))   # Evaluate it at x = 1.0
# prints 0.4199743
```

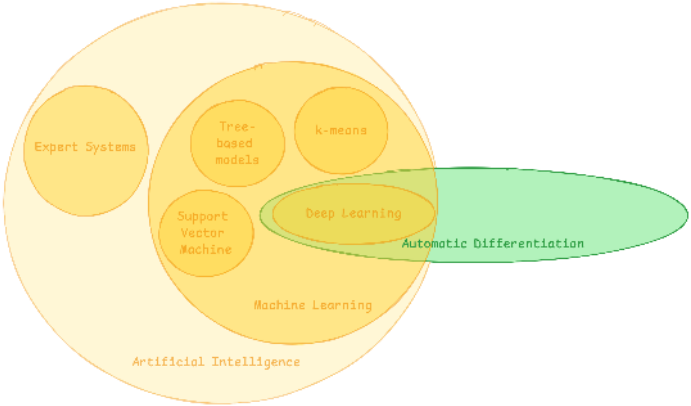
- PyTorch, Tensorflow etc. have their own own math modules
- In these modules, mathematical primitives have defined derivatives
- Code composed with these primitives has derivatives using the chain rule
- Efficient and accurate (Reverse-mode AD $\simeq$ Adjoint method)

- **Example:** $s(a, b) = \sin(a) + ab$



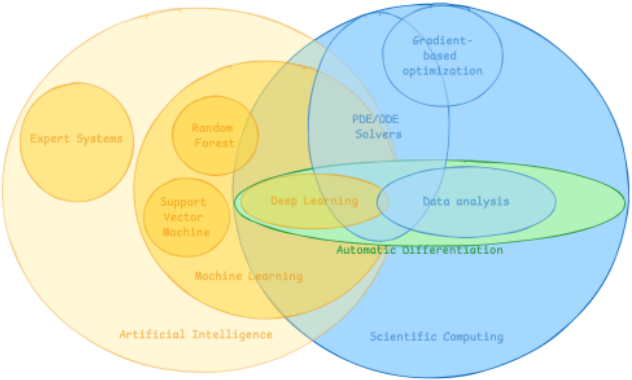
```
from jax import grad
import jax.numpy as jnp

def tanh(x):  # Define a function
  y = jnp.exp(-2.0 * x)
  return (1.0 - y) / (1.0 + y)

grad_tanh = grad(tanh)  # Obtain its gradient function
print(grad_tanh(1.0))   # Evaluate it at x = 1.0
# prints 0.4199743
```

Automatic Differentiation (AD) is intimately related to Deep Learning



Automatic differentiation underpins deep learning. One is not possible without the other.
But AD is not limited to DL!

Automatic Differentiation (AD) is intimately related to Deep Learning AND Scientific Computing

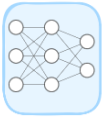


We integrate Deep Learning and Automatic Differentiation with general-purpose Numerical Programming like PDE solvers and Optimization

So how do you bridge this divide?

- Automatic Differentiation
- High-dimensional parametric functions (neural networks)
- (stochastic gradient descent-based) Optimization algorithms
- GPUs

- Decades of computational physics and centuries of mathematical physics knowledge



$$\partial_t f = -v\partial_x f + \ldots$$

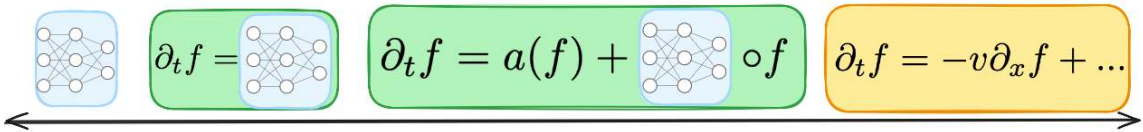
Data-driven

Model-driven

We have a new set of numerical tools, how do we use them?

- Automatic Differentiation
- High-dimensional parametric functions (neural networks)
- (stochastic gradient descent-based) Optimization algorithms
- GPUs

- We do not want to discard decades of computational physics, and centuries of mathematical physics knowledge
- We only seek to replace unknowns with data-driven components



Application: Reduced models for multi-scale simulations - Fluid representation of nonlinear Landau damping[1]

[1] Joglekar and Thomas 2023, *IOP Machine Learning: Science & Technology*.

Enabling multiscale modeling using a hybrid differentiable simulator

Say we want to model linear and nonlinear Landau damping using a fluid code

Enabling multiscale modeling using a hybrid differentiable simulator

Say we want to model linear and nonlinear Landau damping using a fluid code

Kinetic Description

$$\partial_t f + v\,\partial_x f + E_T\,\partial_v f = C_{ee}(f)$$
$$C_{ee}(f) = \nu_{ee}\partial_v(vf + v_0^2\partial_v f)$$

Fluid Description

Poisson Electrostatic Solver

$$E_T = E + E_d(\vec{p})$$
$$\partial_x E = 1 - n_e$$

Enabling multiscale modeling using a hybrid differentiable simulator

Say we want to model linear and nonlinear Landau damping using a fluid code

$$\partial_t f + v\ \partial_x f + E_T\ \partial_v f = C_{ee}(f)$$
$$C_{ee}(f) = \nu_{ee}\partial_v(vf + v_0^2 \partial_v f)$$

$$\partial_t n + \partial_x(u\ n) = 0,$$
$$n\,(\partial_t u + u\ \partial_x u) = -\partial_x p + nE_T,$$
$$p = n^\gamma,$$

$$E_T = E + E_d(\vec{p})$$
$$\partial_x E = 1 - n_e$$

This can't model Landau damping

Enabling multiscale modeling using a hybrid differentiable simulator

Say we want to model linear and nonlinear Landau damping using a fluid code

Kinetic Description

$$\partial_t f + v\ \partial_x f + E_T\ \partial_v f = C_{ee}(f)$$
$$C_{ee}(f) = \nu_{ee}\partial_v(vf + v_0^2\partial_v f)$$

Fluid Description

$$\partial_t n + \partial_x(u\ n) = 0,$$
$$n\,(\partial_t u + u\ \partial_x u) = -\partial_x p + nE_T + 2\ n\ \nu_L * u,$$
$$p = n^\gamma,$$

Poisson Electrostatic Solver

$$E_T = E + E_d(\vec{p})$$
$$\partial_x E = 1 - n_e$$

This can't model nonlinear Landau damping

Enabling multiscale modeling using a hybrid differentiable simulator

Kinetic Description

$$\partial_t f + v\ \partial_x f + E_T\ \partial_v f = C_{ee}(f)$$
$$C_{ee}(f) = \nu_{ee}\partial_v(vf + v_0^2\partial_v f)$$

Relevant Electromagnetic Scales

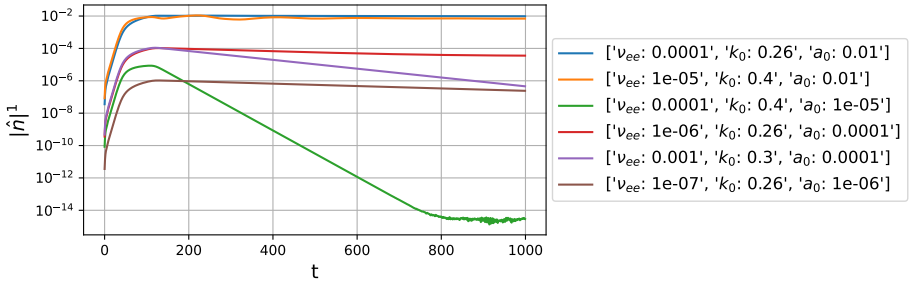
$$E_T = E + E_d(\vec{p})$$
$$\partial_x E = 1 - n_e$$

Fluid Description

$$\partial_t n + \partial_x(u\ n) = 0,$$
$$n\,(\partial_t u + u\ \partial_x u) = -\partial_x p + nE_T + 2\ n\ \frac{\nu_L * u}{1+\delta^2},$$
$$p = n^\gamma,$$
$$\partial_t \delta = v_{ph}\ \partial_x \delta + \nu_g\ \frac{|E * \nu_L|}{1+\delta^2},$$
$$\nu_g = 10^{f(\tilde{E},\tilde{\nu}_{ee},\tilde{k};\theta_g)},$$
$$f(E,\nu_{ee},k;\theta) = 3\tanh\left(g(\tilde{\nu}_{ee},\tilde{k},|\hat{\tilde{E}}|;\theta)\right)$$

This has a quantity that represents trapped particles and can model both

The growth rate NN is trained implicitly via minimizing against kinetic simulations



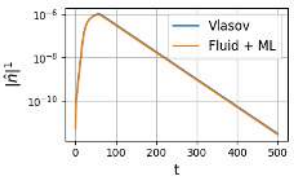
The training data are simulations with parameters given by

$$\nu_{ee} \in [10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}],$$
$$k \in [0.26, 0.28, 0.3, 0.32, 0.34, 0.36, 0.38, 0.4],$$
$$a_0 \in [10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}].$$

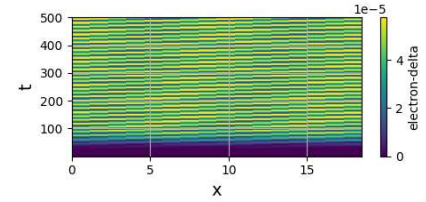
The ML-enhanced fluid model reproduces single-mode NL-EPW

(a)Table stakes - recover non-ML limit

(b) $\delta(t, x)$ stays small



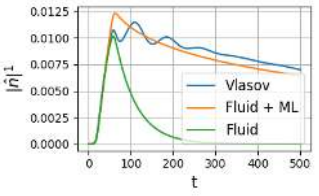
(a) Linear Landau damping



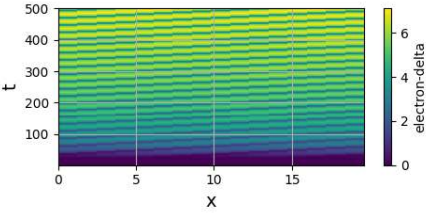
(b) $\delta(t, x)$ shows no trapping

(c) Fluid model damps too fast at the linear value, Fluid + ML recovers Vlasov damping rate

(d) $\delta(t, x)$ grows to ~ 7



(c) Nonlinear Landau damping

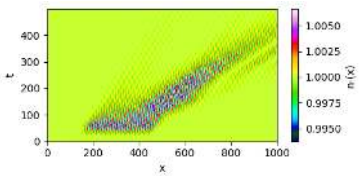


(d) $\delta(t, x)$ is non-zero indicating trapping effects are active

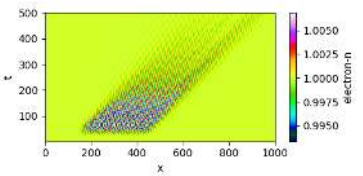
The model is geometry-agnostic and therefore, reproduces finite length behavior without further training

(c) The nonlocal, δ-based model does reproduce etching

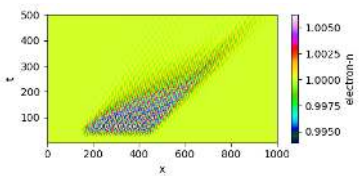
(d) shows that $\delta(t, x)$ grows to account for the hot electron effects and the spatiotemporally nonlocal suppression of Landau damping



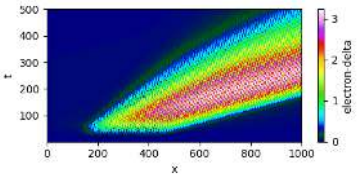
(a) Vlasov



(b) Fluid + Local Model



(c) Fluid + $\delta(t, x)$



(d) $\delta(t, x)$

Application: Finding exotic kinetic dynamics[2]

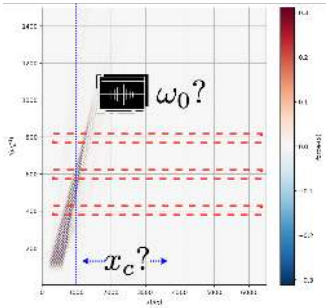
[2] Joglekar and Thomas 2022, *Journal of Plasma Physics*.

Here, the neural network is trained as the forcing function

$$\partial_t f + v\ \partial_x f + E_T\ \partial_v f = C_{ee}(f)$$
$$E_T = E + E_d(\vec{p}(\theta))$$
$$\partial_x E = 1 - n_e$$
$$C_{ee}(f) = \nu_{ee}\partial_v(vf + v_0^2 \partial_v f)$$

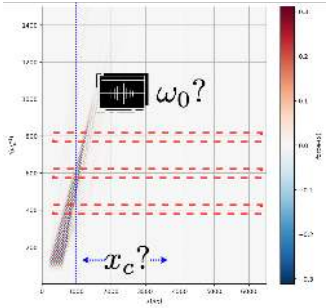
Here, the neural network is trained as the forcing function

$$\partial_t f + v\ \partial_x f + E_T\ \partial_v f = C_{ee}(f)$$
$$E_T = E + E_d(\vec{p}(\theta))$$
$$\partial_x E = 1 - n_e$$
$$C_{ee}(f) = \nu_{ee}\partial_v(vf + v_0^2 \partial_v f)$$



Here, the neural network is trained as the forcing function

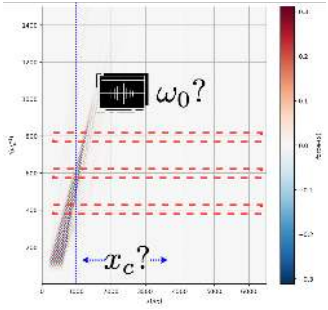
$$\partial_t f + v\ \partial_x f + E_T\ \partial_v f = C_{ee}(f)$$
$$E_T = E + E_d(\vec{p}(\theta))$$
$$\partial_x E = 1 - n_e$$
$$C_{ee}(f) = \nu_{ee}\partial_v(vf + v_0^2 \partial_v f)$$



For $k_0 \in [0.26, 0.27, \ldots, 0.32]$ and $t_c \in [400, 500, \ldots, 800]$, we wish to learn a continuous function for x_c and ω_0

Here, the neural network is trained as the forcing function

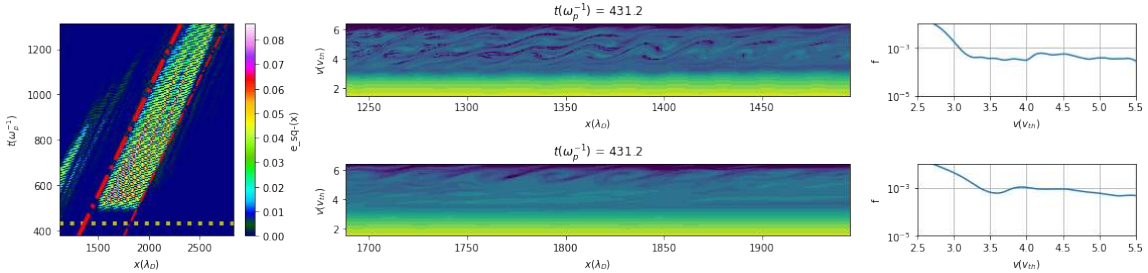
$$\partial_t f + v\ \partial_x f + E_T\ \partial_v f = C_{ee}(f)$$
$$E_T = E + E_d(\vec{p}(\theta))$$
$$\partial_x E = 1 - n_e$$
$$C_{ee}(f) = \nu_{ee}\partial_v(vf + v_0^2 \partial_v f)$$



For $k_0 \in [0.26, 0.27, \ldots, 0.32]$ and $t_c \in [400, 500, \ldots, 800]$, we wish to learn a continuous function for x_c and ω_0

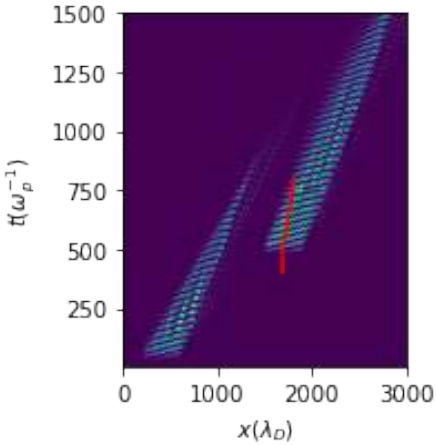
$$\mathcal{L}(\vec{p}(\theta)) = -\sum_{1000}^{1200} \Delta t \sum_{\text{box}} \Delta x \left[E_x^2(t, x; p(\theta)) + \sum_v \Delta v\ (f \log(f) - f_{\text{MX}} \log f_{\text{MX}}) \right]$$

Optimization reveals a technique to create a long-lived wavepacket



- The incoming streaming electrons maintain the flattening of the distribution function
- The flattening of the distribution function prevents the Landau damping at the back of the wavepacket

This phenomenon implies that there is a "critical" inter-speckle distance



- We find a space-time locus at which one can excite long-lived electrostatic plasma waves
- This locus is not at the location of the electrostatic energy but rather, at the region of penetration of the escaped resonant particles
- The distance to the locus is primarily a function of amplitude, wavenumber, and collisionallity

GPU-accelerated AD offers a new set of methods for tackling longstanding problems in plasma physics

- Many problems in plasma physics can be framed as optimization problems
- Modern ML frameworks facilitate general purpose numerical programs that are differentiable
- Writing differentiable programs allows the use of gradient-based techniques to solve those problems
- It also puts neural networks and mathematical modeling on equal footing enabling seamless mixing and matching
- Some example applications are towards
  - **Improving Computational Methods** - Train reduced models of kinetic physics for fluid simulations
  - **Discovering Dynamics** - identify novel nonlinear, kinetic dynamics
  - **Developing Theory** - identify critical points in nonlinear, kinetic physics